

microvg

User Manual



MICROEJ[®]

Reference:	TLT-XXX-MAN-microvg-microvg
Version:	8.0.3
Revision:	XXX

Confidentiality & Intellectual Property

All rights reserved. Information, technical data and tutorials contained in this document are confidential and proprietary under copyright Law of Industrial Smart Software Technology (IS2T S.A.) operating under the brand name MicroEJ®. Without written permission from IS2T S.A., *copying or sending parts of the document or the entire document by any means to third parties is not permitted*. Granted authorizations for using parts of the document or the entire document do not mean IS2T S.A. gives public full access rights.

The information contained herein is not warranted to be error-free. IS2T® and MicroEJ® and all relative logos are trademarks or registered trademarks of IS2T S.A. in France and other Countries.

Java™ is Sun Microsystems' trademark for a technology for developing application software and deploying it in cross-platform, networked environments. When it is used in this documentation without adding the ™ symbol, it includes implementations of the technology by companies other than Sun.

Java™, all Java-based marks and all related logos are trademarks or registered trademarks of Sun Microsystems Inc, in the United States and other Countries.

Other trademarks are proprietary of their authors.

Table of Contents

1. Data Structure Documentation	1
1.1. MICROUI_FontVG Struct Reference	1
1.1.1. Detailed Description	1
1.2. MICROVG_PATH_HEADER Struct Reference	1
1.2.1. Detailed Description	2
1.2.2. Field Documentation	2
1.3. VG_OUTLINE_BOX Struct Reference	3
1.3.1. Detailed Description	4
1.3.2. Field Documentation	4
2. File Documentation	6
2.1. bsp/vg/inc/ui_drawing_bvi.h File Reference	6
2.1.1. Detailed Description	6
2.2. bsp/vg/inc/vg_configuration.h File Reference	6
2.2.1. Detailed Description	8
2.2.2. Macro Definition Documentation	8
2.3. bsp/vg/inc/vg_drawing.h File Reference	11
2.3.1. Detailed Description	12
2.3.2. Function Documentation	12
2.4. bsp/vg/inc/vg_drawing_bvi.h File Reference	17
2.4.1. Detailed Description	17
2.5. bsp/vg/inc/vg_drawing_stub.h File Reference	17
2.5.1. Detailed Description	18
2.6. bsp/vg/inc/vg_freetype.h File Reference	18
2.6.1. Detailed Description	19
2.6.2. Typedef Documentation	19
2.6.3. Function Documentation	20
2.7. bsp/vg/inc/vg_helper.h File Reference	22
2.7.1. Detailed Description	24
2.7.2. Macro Definition Documentation	24
2.7.3. Function Documentation	24
2.8. bsp/vg/inc/vg_outline_box.h File Reference	29
2.8.1. Detailed Description	30
2.8.2. Function Documentation	30
2.9. bsp/vg/inc/vg_path.h File Reference	30
2.9.1. Detailed Description	31
2.9.2. Function Documentation	32
2.10. bsp/vg/src/LLVG_BVI_impl_stub.c File Reference	35
2.10.1. Detailed Description	35
2.11. bsp/vg/src/LLVG_FONT_impl_freetype.c File Reference	35
2.11.1. Detailed Description	36
2.12. bsp/vg/src/LLVG_FONT_impl_stub.c File Reference	36
2.12.1. Detailed Description	36
2.13. bsp/vg/src/LLVG_GRADIENT_impl_stub.c File Reference	36
2.13.1. Detailed Description	37

2.14. bsp/vg/src/LLVG_MATRIX_impl.c File Reference	37
2.14.1. Detailed Description	38
2.15. bsp/vg/src/LLVG_PAINTER_impl.c File Reference	38
2.15.1. Detailed Description	38
2.16. bsp/vg/src/LLVG_PATH_impl_dual.c File Reference	39
2.16.1. Detailed Description	39
2.17. bsp/vg/src/LLVG_PATH_impl_single.c File Reference	39
2.17.1. Detailed Description	39
2.18. bsp/vg/src/LLVG_PATH_impl_stub.c File Reference	40
2.18.1. Detailed Description	40
2.19. bsp/vg/src/vg_freetype_path.c File Reference	40
2.19.1. Detailed Description	40
2.20. bsp/vg/src/vg_helper.c File Reference	40
2.20.1. Detailed Description	42
2.20.2. Function Documentation	42
2.21. bsp/vg/src/vg_outline_box.c File Reference	45
2.21.1. Detailed Description	45
Index	46

List of Tables

2.1.	12
2.2.	13
2.3.	13
2.4.	14
2.5.	14
2.6.	15
2.7.	16
2.8.	16
2.9.	17
2.10.	20
2.11.	20
2.12.	21
2.13.	22
2.14.	25
2.15.	26
2.16.	26
2.17.	27
2.18.	28
2.19.	28
2.20.	29
2.21.	30
2.22.	32
2.23.	32
2.24.	33
2.25.	33
2.26.	34
2.27.	34
2.28.	34
2.29.	42
2.30.	43
2.31.	44
2.32.	45

Chapter 1. Data Structure Documentation

1.1. MICROUI_FontVG Struct Reference

Represents a MicroUI Font extended by MicroVG.

Data Fields

- MICROUI_Font ui_font
The MicroUI Font header.
- jint vg_font_handle
The identifier of the MicroVG Font.
- jint vg_font_size
The size to apply on the MicroVG Font.

1.1.1. Detailed Description

Represents a MicroUI Font extended by MicroVG.

See also. `getFont(int size)` in `VectorFont.java`

Definition at line 67 of file `ui_font_drawing_vg.c`.

The documentation for this struct was generated from the following file:
`bsp/vg/src/ui_font_drawing_vg.c`

1.2. MICROVG_PATH_HEADER Struct Reference

Map a jbyte array that represents a path.

```
#include <vg_path.h>
```

Data Fields

- uint32_t data_size
- uint8_t
- uint8_t padding1
- uint8_t padding2
- uint8_t padding3

- float bounds_xmin
- float bounds_xmax
- float bounds_ymin
- float bounds_ymax

1.2.1. Detailed Description

Map a jbyte array that represents a path.

Definition at line 54 of file vg_path.h.

1.2.2. Field Documentation

1.2.2.1. bounds_xmax

```
float MICROVG_PATH_HEADER::bounds_xmax
```

right

Definition at line 67 of file vg_path.h.

1.2.2.2. bounds_xmin

```
float MICROVG_PATH_HEADER::bounds_xmin
```

left

Definition at line 66 of file vg_path.h.

1.2.2.3. bounds_ymax

```
float MICROVG_PATH_HEADER::bounds_ymax
```

bottom

Definition at line 69 of file vg_path.h.

1.2.2.4. bounds_ymin

```
float MICROVG_PATH_HEADER::bounds_ymin
```

top

Definition at line 68 of file vg_path.h.

1.2.2.5. data_size

`uint32_t MICROVG_PATH_HEADER::data_size`

data size (without header)

Definition at line 56 of file `vg_path.h`.

1.2.2.6. padding1

`uint8_t MICROVG_PATH_HEADER::padding1`

Padding, unused.

Definition at line 63 of file `vg_path.h`.

1.2.2.7. padding2

`uint8_t MICROVG_PATH_HEADER::padding2`

Padding, unused.

Definition at line 64 of file `vg_path.h`.

1.2.2.8. padding3

`uint8_t MICROVG_PATH_HEADER::padding3`

Padding, unused.

Definition at line 65 of file `vg_path.h`.

The documentation for this struct was generated from the following file:
`bsp/vg/inc/vg_path.h`

1.3. VG_OUTLINE_BOX Struct Reference

The box of the curve an outline draws, control points excluded.

```
#include <vg_outline_box.h>
```

Data Fields

- `float x_min`
- `float x_max`
- `float y_min`
- `float y_max`
- `bool empty`

1.3.1. Detailed Description

The box of the curve an outline draws, control points excluded.

Coordinates are in the units of the outline they were computed from. Every caller loads its glyphs with FT_LOAD_NO_SCALE, so they are font units.

Definition at line 47 of file vg_outline_box.h.

1.3.2. Field Documentation

1.3.2.1. empty

```
bool VG_OUTLINE_BOX::empty
```

true when the outline draws nothing: the four coordinates are then 0

Definition at line 52 of file vg_outline_box.h.

1.3.2.2. x_max

```
float VG_OUTLINE_BOX::x_max
```

The largest x coordinate of the box.

Definition at line 49 of file vg_outline_box.h.

1.3.2.3. x_min

```
float VG_OUTLINE_BOX::x_min
```

The smallest x coordinate of the box.

Definition at line 48 of file vg_outline_box.h.

1.3.2.4. y_max

```
float VG_OUTLINE_BOX::y_max
```

The largest y coordinate of the box.

Definition at line 51 of file vg_outline_box.h.

1.3.2.5. y_min

```
float VG_OUTLINE_BOX::y_min
```

The smallest y coordinate of the box.

Definition at line 50 of file `vg_outline_box.h`.

The documentation for this struct was generated from the following file:
`bsp/vg/inc/vg_outline_box.h`

Chapter 2. File Documentation

2.1. bsp/vg/inc/ui_drawing_bvi.h File Reference

Implementation of a set of ui_drawing.h drawing functions (MicroUI and Drawing libraries). These are implementations over a BufferedVectorImage and the destination buffer format is the BufferedVectorImage format. When the drawing cannot be performed for any reason, the stub implementation is used instead.

```
#include "vg_configuration.h"
```

2.1.1. Detailed Description

Implementation of a set of ui_drawing.h drawing functions (MicroUI and Drawing libraries). These are implementations over a BufferedVectorImage and the destination buffer format is the BufferedVectorImage format. When the drawing cannot be performed for any reason, the stub implementation is used instead.

Author. MicroEJ Developer Team

Version. 8.0.3

2.2. bsp/vg/inc/vg_configuration.h File Reference

MicroEJ MicroVG library low level API: enable some features according to the hardware capacities.

```
#include "veeport_configuration.h"
#include "ui_configuration.h"
#include <stdio.h>
```

Macros

- #define VG_FEATURE_PATH_SINGLE_ARRAY (1)

Value of "VG_FEATURE_PATH" to use only one array to store the path's commands and the commands' parameters.

- #define VG_FEATURE_PATH_DUAL_ARRAY (2)

Value of "VG_FEATURE_PATH" to use two arrays to store the path's data: one for the commands and one for the commands' parameters.

- #define VG_FEATURE_GRADIENT_FULL (1)

Value of "VG_FEATURE_GRADIENT" to use a full implementation of the MicroVG's LinearGradient.

- `#define VG_FEATURE_GRADIENT_FIRST_COLOR (2)`
Value of "VG_FEATURE_GRADIENT" to use a reduced implementation of the MicroVG's Linear-Gradient.
- `#define VG_FEATURE_FONT_FREETYPE_VECTOR (1)`
Value of "VG_FEATURE_FONT" to use FreeType as implementation of the MicroVG's VectorFont.
- `#define VG_FEATURE_FONT_FREETYPE_BITMAP (2)`
Value of "VG_FEATURE_FONT" to use FreeType as implementation of the MicroVG's VectorFont.
- `#define VG_FEATURE_PATH VG_FEATURE_PATH_SINGLE_ARRAY`
Set this define to embed the implementation of the MicroVG's Path (dynamic path creation and path rendering).
- `#define VG_FEATURE_GRADIENT VG_FEATURE_GRADIENT_FULL`
Set this define to specify the implementation of the MicroVG's LinearGradient (dynamic gradient creation and drawings with gradient).
- `#define VG_FEATURE_FONT VG_FEATURE_FONT_FREETYPE_VECTOR`
Set this define to specify the implementation of the MicroVG's VectorFont (dynamic font loading and text rendering).
- `#define VG_FEATURE_FREETYPE_TTF (0)`
Set this define to 1 to enable the support of TTF font files (disabled by default).
- `#define VG_FEATURE_FREETYPE_OTF (0)`
Set this define to 1 to enable the support of OTF font files (disabled by default).
- `#define VG_FEATURE_FREETYPE_COLORED_EMOJI (0)`
Set this define to 1 to enable the support of colored emoji (disabled by default).
- `#define VG_FEATURE_FONT_COMPLEX_LAYOUT (0)`
Set this define to 1 to enable the support of complex layout (disabled by default).
- `#define VG_FEATURE_FONT_EXTERNAL (0)`
Set this define to 1 to enable the support of external font files (disabled by default).
- `#define VG_FEATURE_FREETYPE_HEAP_SIZE (80 * 1024)`
Configure this define to set the freetype heap size.
- `#define VG_LOG_INFO_ENABLED (0)`
Configure this define to set the complex layouter heap size.
- `#define VG_LOG_DECLARE_MODULE()`
Declares the log module, once at the top of each source file that logs.

- `#define VG_LOG_ERROR_PRINT(fmt, ...) (void)printf("[ERROR - MICROVG] : " fmt "\n", ##__VA_ARGS__)`

Prints an error log through "printf". The format has no trailing line return.

- `#define VG_LOG_INFO_PRINT(fmt, ...) (void)printf("[INFO - MICROVG] : " fmt "\n", ##__VA_ARGS__)`

Prints an informative log through "printf". The format has no trailing line return.

2.2.1. Detailed Description

MicroEJ MicroVG library low level API: enable some features according to the hardware capacities.

Author. MicroEJ Developer Team

2.2.2. Macro Definition Documentation

2.2.2.1. VG_FEATURE_FONT

```
#define VG_FEATURE_FONT VG_FEATURE_FONT_FREETYPE_VECTOR
```

Set this define to specify the implementation of the MicroVG' VectorFont (dynamic font loading and text rendering).

See also. `VG_FEATURE_FONT_FREETYPE_VECTOR VG_FEATURE_FONT_FREETYPE_BITMAP`

When not set, a stub implementation is used. The application cannot load a font (and by consequence cannot draw a text).

Definition at line 144 of file `vg_configuration.h`.

2.2.2.2. VG_FEATURE_FONT_COMPLEX_LAYOUT

```
#define VG_FEATURE_FONT_COMPLEX_LAYOUT (0)
```

Set this define to 1 to enable the support of complex layout (disabled by default).

When set, the complex layout feature is disabled by default (the Freetype layout manager is used). See functions `MICROVG_HELPER_set_complex_layout()` and `MICROVG_HELPER_has_complex_layouter()`.

Note: the complex layout feature is managed by the Harfbuzz engine. Harfbuzz is used beside Freetype, thus the `FT_CONFIG_OPTION_USE_HARFBUZZ` define is not needed. This implementation has been chosen to ease the replacement of harfbuzz by an other complex layouter.

Definition at line 181 of file `vg_configuration.h`.

2.2.2.3. VG_FEATURE_FONT_EXTERNAL

```
#define VG_FEATURE_FONT_EXTERNAL (0)
```

Set this define to 1 to enable the support of external font files (disabled by default).

When a font file is not available in the application classpath, the implementation tries to load it from an external resource system.

The VEE Port must embed the module "External Resource" and the BSP must implement "LLEXTERES_impl.h" header file.

When not set, only the resources compiled with the application are used.

Definition at line 196 of file vg_configuration.h.

2.2.2.4. VG_FEATURE_FONT_FREETYPE_BITMAP

```
#define VG_FEATURE_FONT_FREETYPE_BITMAP (2)
```

Value of "VG_FEATURE_FONT" to use FreeType as implementation of the MicroVG's VectorFont.

This implementation configures FreeType to manipulate the font's glyphs as bitmaps. No renderer is required

Definition at line 97 of file vg_configuration.h.

2.2.2.5. VG_FEATURE_FONT_FREETYPE_VECTOR

```
#define VG_FEATURE_FONT_FREETYPE_VECTOR (1)
```

Value of "VG_FEATURE_FONT" to use FreeType as implementation of the MicroVG's VectorFont.

This implementation configures FreeType to manipulate the font's glyphs as a succession of vector paths. It requires a "vector" renderer.

Definition at line 88 of file vg_configuration.h.

2.2.2.6. VG_FEATURE_FREETYPE_HEAP_SIZE

```
#define VG_FEATURE_FREETYPE_HEAP_SIZE (80 * 1024)
```

Configure this define to set the freetype heap size.

The freetype heap size depends on the font used by the application

See also. MICROVG_MONITOR_HEAP in vg_helper.h to monitor the heap usage evolution.

Definition at line 206 of file vg_configuration.h.

2.2.2.7. VG_FEATURE_GRADIENT

```
#define VG_FEATURE_GRADIENT VG_FEATURE_GRADIENT_FULL
```

Set this define to specify the implementation of the MicroVG's LinearGradient (dynamic gradient creation and drawings with gradient).

See also. VG_FEATURE_GRADIENT_FULL VG_FEATURE_GRADIENT_FIRST_COLOR

When not set, a stub implementation is used. No error is thrown at runtime when the application uses a gradient: the dynamic gradient are not created and the gradient rendering is not performed.

Definition at line 130 of file `vg_configuration.h`.

2.2.2.8. VG_FEATURE_GRADIENT_FIRST_COLOR

```
#define VG_FEATURE_GRADIENT_FIRST_COLOR (2)
```

Value of "VG_FEATURE_GRADIENT" to use a reduced implementation of the MicroVG's LinearGradient.

This implementation keeps only the first color set by the application when inializing a gradient.

Definition at line 75 of file `vg_configuration.h`.

2.2.2.9. VG_FEATURE_GRADIENT_FULL

```
#define VG_FEATURE_GRADIENT_FULL (1)
```

Value of "VG_FEATURE_GRADIENT" to use a full implementation of the MicroVG's LinearGradient.

This implementation holds an array of all colors and positions set by the application.

Definition at line 66 of file `vg_configuration.h`.

2.2.2.10. VG_FEATURE_PATH

```
#define VG_FEATURE_PATH VG_FEATURE_PATH_SINGLE_ARRAY
```

Set this define to embed the implementation of the MicroVG's Path (dynamic path creation and path rendering).

This implementation holds the path's commands and paramters defined by the application. The define value specifies how this data is stored (one or two arrays).

When not set, a stub implementation is used. No error is thrown at runtime when the application uses a path: the dynamic path are not created and the path rendering is not performed.

Definition at line 115 of file `vg_configuration.h`.

2.2.2.11. VG_LOG_DECLARE_MODULE

```
#define VG_LOG_DECLARE_MODULE( )
```

Declares the log module, once at the top of each source file that logs.

Empty: "printf" needs no declaration.

Definition at line 320 of file `vg_configuration.h`.

2.2.2.12. VG_LOG_INFO_ENABLED

```
#define VG_LOG_INFO_ENABLED (0)
```

Configure this define to set the complex layouter heap size.

See also. `VG_FEATURE_FONT_COMPLEX_LAYOUT`

The complex layouter heap size depends on the font used by the application

See also. `MICROVG_MONITOR_HEAP` in `vg_helper.h` to monitor the heap usage evolution.

Set this define to 1 to compile the informative logs (disabled by default).

The error logs are always compiled.

Definition at line 256 of file `vg_configuration.h`.

2.3. bsp/vg/inc/vg_drawing.h File Reference

Provides drawing functions called by MicroVG native drawing functions.

```
#include <LLVG_PAINTER_impl.h>
#include "ui_drawing.h"
```

Functions

- `void VG_DRAWING_initialize (void)`

Called when initializing MicroVG.

- `bool VG_DRAWING_image_is_closed (const MICROVG_Image *image)`

Tells if the image is closed (closed just before calling the native function).

- `void VG_DRAWING_get_image_size (const MICROVG_Image *image, float *width, float *height)`

Gets the image size. The image structure format is VEE Port specific.

- `DRAWING_Status VG_DRAWING_drawPath (MICROUI_GraphicsContext *gc, jbyte *path, jfloat *matrix, jint fillRule, jint blend, jint color)`

Draws a path filled with the specified color (not the one of the Graphics Context).

- `DRAWING_Status VG_DRAWING_drawGradient (MICROUI_GraphicsContext *gc, jbyte *path, jfloat *matrix, jint fillRule, jint alpha, jint blend, jint *gradient, jfloat *gradientMatrix)`

Draws a path filled with a linear gradient.

- `DRAWING_Status VG_DRAWING_drawString (MICROUI_GraphicsContext *gc, jchar *text, jint length, jint faceHandle, jfloat size, jfloat *matrix, jint alpha, jint blend, jfloat letterSpacing)`

Draws a string with the color of the Graphics Context.

- `DRAWING_Status VG_DRAWING_drawStringGradient (MICROUI_GraphicsContext *gc, jchar *text, jint length, jint faceHandle, jfloat size, jfloat *matrix, jint alpha, jint blend, jfloat letterSpacing, jint *gradientData, jfloat *gradientMatrix)`

Draws a string with a linear gradient.

- DRAWING_Status VG_DRAWING_drawStringOnCircle (MICROUI_GraphicsContext *gc, jchar *text, jint length, jint faceHandle, jfloat size, jfloat *matrix, jint alpha, jint blend, jfloat letterSpacing, jfloat radius, jint direction)

Draws a string along a circle, with the color of the Graphics Context.

- DRAWING_Status VG_DRAWING_drawStringOnCircleGradient (MICROUI_GraphicsContext *gc, jchar *text, jint length, jint faceHandle, jfloat size, jfloat *matrix, jint alpha, jint blend, jfloat letterSpacing, jfloat radius, jint direction, jint *gradientData, jfloat *gradientMatrix)

Draws a string along a circle, with a linear gradient.

- DRAWING_Status VG_DRAWING_drawImage (MICROUI_GraphicsContext *gc, void *image, jfloat *matrix, jint alpha, jlong elapsed, const float color_matrix[], jint *errno)

Draws an image with transformation and opacity. Optionally apply an animation and / or a color filtering.

2.3.1. Detailed Description

Provides drawing functions called by MicroVG native drawing functions.

This header file follows strictly the same concept as ui_drawing.h but for MicroVG library.

See also. `ui_drawing.h`

2.3.2. Function Documentation

2.3.2.1. VG_DRAWING_drawGradient()

```
DRAWING_Status VG_DRAWING_drawGradient (MICROUI_GraphicsContext * gc,  
jbyte * path, jfloat * matrix, jint fillRule, jint alpha, jint blend,  
jint * gradient, jfloat * gradientMatrix)
```

Draws a path filled with a linear gradient.

Parameters .

Table 2.1.

in	gc	the destination
in	path	the path's data
in	matrix	the deformation to apply
in	fillRule	the fill type
in	alpha	the global opacity to apply
in	blend	the blend mode
in	gradient	the gradient to apply

in	gradientMatrix	the gradient deformation, may be null (means identity)
----	----------------	--

Returns. the drawing status.

2.3.2.2. VG_DRAWING_drawImage()

```
DRAWING_Status VG_DRAWING_drawImage (MICROUI_GraphicsContext * gc, void
* image, jfloat * matrix, jint alpha, jlong elapsed, const float
color_matrix[], jint * errno)
```

Draws an image with transformation and opacity. Optionally apply an animation and / or a color filtering.

Parameters .

Table 2.2.

in	gc	the graphics context to draw on
in	image	the image to draw
in	matrix	the transformation matrix to apply
in	alpha	the global opacity rendering value
in	elapsed	the elapsed time since the beginning of the animation, in milliseconds
in	color_matrix	the color matrix used to transform colors, may be null (means no color deformation)
out	errno	the error code to return to MicroVG.

Returns. the drawing status.

2.3.2.3. VG_DRAWING_drawPath()

```
DRAWING_Status VG_DRAWING_drawPath (MICROUI_GraphicsContext * gc, jbyte
* path, jfloat * matrix, jint fillRule, jint blend, jint color)
```

Draws a path filled with the specified color (not the one of the Graphics Context).

Parameters .

Table 2.3.

in	gc	the destination
in	path	the path's data
in	matrix	the deformation to apply
in	fillRule	the fill type
in	blend	the blend mode
in	color	the color to apply (with an opacity)

Returns. the drawing status.

2.3.2.4. VG_DRAWING_drawString()

```
DRAWING_Status VG_DRAWING_drawString (MICROUI_GraphicsContext * gc,  
jchar * text, jint length, jint faceHandle, jfloat size, jfloat * matrix,  
jint alpha, jint blend, jfloat letterSpacing)
```

Draws a string with the color of the Graphics Context.

The top/left position of the first drawn character is placed at coordinate x/y.

Parameters .

Table 2.4.

in	gc	the MicroUI GraphicsContext target.
in	text	the array of characters to draw.
in	length	the length of the array
in	faceHandle	the font reference handle.
in	size	the height of the font in pixels.
in	matrix	deformation matrix, may be null (means identity)
in	alpha	the opacity level to apply to the character.
in	blend	the blend mode to use
in	letterSpacing	the extra letter spacing to use

Returns. the drawing status.

2.3.2.5. VG_DRAWING_drawStringGradient()

```
DRAWING_Status VG_DRAWING_drawStringGradient (MICROUI_GraphicsContext *  
gc, jchar * text, jint length, jint faceHandle, jfloat size, jfloat *  
matrix, jint alpha, jint blend, jfloat letterSpacing, jint * gradient-  
Data, jfloat * gradientMatrix)
```

Draws a string with a linear gradient.

Parameters .

Table 2.5.

in	gc	the MicroUI GraphicsContext target.
in	text	the array of characters to draw.
in	length	the length of the array
in	faceHandle	the font reference handle.
in	size	the height of the font in pixels.

in	matrix	deformation matrix, may be null (means identity)
in	alpha	the opacity level to apply to the character.
in	blend	the blend mode to use
in	letterSpacing	the extra letter spacing to use
in	gradientData	the gradient to apply
in	gradientMatrix	the gradient deformation, may be null (means identity)

Returns. the drawing status.

2.3.2.6. VG_DRAWING_drawStringOnCircle()

```
DRAWING_Status VG_DRAWING_drawStringOnCircle (MICROUI_GraphicsContext *
gc, jchar * text, jint length, jint faceHandle, jfloat size, jfloat *
matrix, jint alpha, jint blend, jfloat letterSpacing, jfloat radius,
jint direction)
```

Draws a string along a circle, with the color of the Graphics Context.

Parameters .

Table 2.6.

in	gc	the MicroUI GraphicsContext target.
in	text	the array of characters to draw.
in	length	the length of the array
in	faceHandle	the font reference handle.
in	size	the height of the font in pixels.
in	matrix	deformation matrix, may be null (means identity)
in	alpha	the opacity level to apply to the character.
in	blend	the blend mode to use
in	letterSpacing	the extra letter spacing to use
in	radius	the radius of the circle
in	direction	the direction of the text along the circle

Returns. the drawing status.

2.3.2.7. VG_DRAWING_drawStringOnCircleGradient()

```
DRAWING_Status VG_DRAWING_drawStringOnCircleGradient
(MICROUI_GraphicsContext * gc, jchar * text, jint length, jint faceHan-
dle, jfloat size, jfloat * matrix, jint alpha, jint blend, jfloat let-
terSpacing, jfloat radius, jint direction, jint * gradientData, jfloat
* gradientMatrix)
```

Draws a string along a circle, with a linear gradient.

Parameters .

Table 2.7.

in	gc	the MicroUI GraphicsContext target.
in	text	the array of characters to draw.
in	length	the length of the array
in	faceHandle	the font reference handle.
in	size	the height of the font in pixels.
in	matrix	deformation matrix, may be null (means identity)
in	alpha	the opacity level to apply to the character.
in	blend	the blend mode to use
in	letterSpacing	the extra letter spacing to use
in	radius	the radius of the circle
in	direction	the direction of the text along the circle
in	gradientData	the gradient to apply
in	gradientMatrix	the gradient deformation, may be null (means identity)

Returns. the drawing status.

2.3.2.8. VG_DRAWING_get_image_size()

```
void VG_DRAWING_get_image_size (const MICROVG_Image * image, float *  
width, float * height)
```

Gets the image size. The image structure format is VEE Port specific.

Parameters .

Table 2.8.

in	image	the image
out	width	the image width
out	height	the image height

2.3.2.9. VG_DRAWING_image_is_closed()

```
bool VG_DRAWING_image_is_closed (const MICROVG_Image * image)
```

Tells if the image is closed (closed just before calling the native function).

Parameters .

Table 2.9.

in	image	the image to check.
----	-------	---------------------

Returns. true if the image is closed.

2.3.2.10. VG_DRAWING_initialize()

```
void VG_DRAWING_initialize (void )
```

Called when initializing MicroVG.

This function should initialize the necessary variables.

2.4. bsp/vg/inc/vg_drawing_bvi.h File Reference

Implementation of a set of `vg_drawing.h` drawing functions (library MicroVG). These are implementations over a `BufferedImage` and the destination buffer format is the `BufferedImage` format. When the drawing cannot be performed for any reason, the stub implementation is used instead.

```
#include "vg_configuration.h"
```

2.4.1. Detailed Description

Implementation of a set of `vg_drawing.h` drawing functions (library MicroVG). These are implementations over a `BufferedImage` and the destination buffer format is the `BufferedImage` format. When the drawing cannot be performed for any reason, the stub implementation is used instead.

Author. MicroEJ Developer Team

Version. 8.0.3

See also. `ui_drawing_bvi.h`

2.5. bsp/vg/inc/vg_drawing_stub.h File Reference

Implementation of all drawing functions of `vg_drawing.h` (library MicroVG). These are stubbed implementations, there is no error but the drawings are not performed. Useful to stub a drawing on a custom destination (custom `GraphicsContext` format not supported by the Graphics Engine).

```
#include "vg_drawing.h"
```

Functions

- `DRAWING_Status` (`MICROUI_GraphicsContext *gc`, `jbyte *path`, `jfloat *matrix`, `jint fillRule`, `jint blend`, `jint color`)

- DRAWING_Status (MICROUI_GraphicsContext *gc, jbyte *path, jfloat *matrix, jint fillRule, jint alpha, jint blend, jint *gradient, jfloat *gradientMatrix)
- DRAWING_Status (MICROUI_GraphicsContext *gc, jchar *text, jint length, jint faceHandle, jfloat size, jfloat *matrix, jint alpha, jint blend, jfloat letterSpacing)
- DRAWING_Status (MICROUI_GraphicsContext *gc, jchar *text, jint length, jint faceHandle, jfloat size, jfloat *matrix, jint alpha, jint blend, jfloat letterSpacing, jint *gradientData, jfloat *gradientMatrix)
- DRAWING_Status (MICROUI_GraphicsContext *gc, jchar *text, jint length, jint faceHandle, jfloat size, jfloat *matrix, jint alpha, jint blend, jfloat letterSpacing, jfloat radius, jint direction)
- DRAWING_Status (MICROUI_GraphicsContext *gc, jchar *text, jint length, jint faceHandle, jfloat size, jfloat *matrix, jint alpha, jint blend, jfloat letterSpacing, jfloat radius, jint direction, jint *gradientData, jfloat *gradientMatrix)
- DRAWING_Status (MICROUI_GraphicsContext *gc, void *image, jfloat *matrix, jint alpha, jlong elapsed, const float color_matrix[], jint *errno)

2.5.1. Detailed Description

Implementation of all drawing functions of `vg_drawing.h` (library MicroVG). These are stubbed implementations, there is no error but the drawings are not performed. Useful to stub a drawing on a custom destination (custom GraphicsContext format not supported by the Graphics Engine).

Author. MicroEJ Developer Team

Version. 8.0.3

See also. `ui_drawing_stub.h`

2.6. bsp/vg/inc/vg_freetype.h File Reference

MicroEJ MicroVG library low level API: implementation over FreeType.

```
#include "vg_configuration.h"
#include "vg_path.h"
```

Typedefs

- typedef jint(* VG_FREETYPE_draw_glyph_t) (VG_PATH_HEADER_t *path, jfloat *matrix, uint32_t color, bool fill_rule_even_odd, void *user_data)

Function to draw a character element: a glyph. A glyph is a vectorial path.

Functions

- void VG_FREETYPE_initialize (void)

Initializes the lowlevel font library.

- `bool VG_FREETYPE_string_span (const jchar *text, jint length, jint face_handle, jfloat letter_spacing_font_units, jfloat *left, jfloat *right)`

Folds the ink of a string and returns its two extreme edges.

- `jfloat VG_FREETYPE_string_width (jchar *text, jint length, jint face_handle, jfloat size, jfloat letter_spacing)`

Measures the width of a text for the specified font and size.

- `jfloat VG_FREETYPE_get_scale (jint face_handle, jfloat size)`

Computes the scale that converts font units to pixels at the given size.

- `jint VG_FREETYPE_draw_string (VG_FREETYPE_draw_glyph_t drawer, const jchar *text, jint length, jint face_handle, jfloat size, const jfloat *matrix, uint32_t color, jfloat letter_spacing, jfloat radius, jint direction, void *user_data)`

Draws a string using the FreeType engine along a line or a circle, with a color or a linear gradient. The implementation does not draw, it calls the drawer function for each glyph: `VG_FREETYPE_draw_glyph_t`.

- `jint (jint color)`

2.6.1. Detailed Description

MicroEJ MicroVG library low level API: implementation over FreeType.

Author. MicroEJ Developer Team

Version. 8.0.3

2.6.2. Typedef Documentation

2.6.2.1. VG_FREETYPE_draw_glyph_t

```
typedef jint(* VG_FREETYPE_draw_glyph_t) ( VG_PATH_HEADER_t *path,
jfloat *matrix, uint32_t color, bool fill_rule_even_odd, void
*user_data)
```

Function to draw a character element: a glyph. A glyph is a vectorial path.

This function is called by `VG_FREETYPE_draw_string()`. Two implementations are required: one that draws the path with a color and one that draws the path with a gradient.

The list of the drawing parameters is reduced to the elements that the FreeType engine changes for each glyph: the glyph's path, the transformation to apply on the glyph, the glyph color (useful for colored glyphs like emoji), the glyph's outline.

All others elements required by the drawer (if any) have to be listed in the dedicated structure `user_data`.

Returns. : the MicroVG error code

Definition at line 56 of file vg_freetype.h.

2.6.3. Function Documentation

2.6.3.1. VG_FREETYPE_draw_string()

```
jint VG_FREETYPE_draw_string (VG_FREETYPE_draw_glyph_t drawer, const
jchar * text, jint length, jint face_handle, jfloat size, const jfloat
* matrix, uint32_t color, jfloat letter_spacing, jfloat radius, jint
direction, void * user_data)
```

Draws a string using the Freetype engine along a line or a circle, with a color or a linear gradient. The implementation does not draw, it calls the drawer function for each glyph: VG_FREETYPE_draw_glyph_t.

Parameters .

Table 2.10.

in	drawer	the function to draw a glyph.
in	text	the array of characters to draw.
in	length	the length of the array
in	face_handle	the font reference handle.
in	size	the height of the font in pixels.
in	matrix	the transformation to apply.
in	color	the 32-bit color to apply (useless when drawing with a gradient).
in	letter_spacing	the extra letter spacing to use.
in	radius	the radius of the circle (0 to draw along a line).
in	direction	the direction of the text along the circle (0 to draw along a line).
in	user_data	the data used by the drawer; may be null.

Returns. LLVG_SUCCESS if something has been drawn, an different value otherwise

2.6.3.2. VG_FREETYPE_get_scale()

```
jfloat VG_FREETYPE_get_scale (jint face_handle, jfloat size)
```

Computes the scale that converts font units to pixels at the given size.

LLVG_FONT_IMPL_load_font() refuses a face whose EM size is 0, so the division is always defined.

Parameters .

Table 2.11.

in	face_handle	the font reference handle.
----	-------------	----------------------------

in	size	the height of the font in pixels.
----	------	-----------------------------------

Returns. the size of one font unit, in pixels.

2.6.3.3. VG_FREETYPE_string_span()

```
bool VG_FREETYPE_string_span (const jchar * text, jint length, jint
face_handle, jfloat letter_spacing_font_units, jfloat * left, jfloat *
right)
```

Folds the ink of a string and returns its two extreme edges.

Every glyph is placed at its own pen position, and a glyph that draws nothing contributes no edge at either end. The pen carries the letter spacing, so a string that draws no glyph has no span whatever the spacing is.

Returns the two extreme edges of the ink; the caller decides what to do with them.

A caller that passes NULL for the right edge asks for the left one alone. The fold then stops as soon as no remaining glyph can reach further left, so it reads a few glyphs instead of the whole string. It returns the same left edge as a full fold as long as no glyph is placed more than one EM to the left of its own pen position, which is the margin the stop keeps; a letter spacing below zero and an advance seen to be negative each disable the stop instead of narrowing that margin.

The function leaves no iteration open, so a caller that iterates afterwards must call VG_HELPER_layout_configure() again.

Parameters .

Table 2.12.

in	text	the array of characters to fold.
in	length	the length of the array.
in	face_handle	the font reference handle.
in	letter_spacing_font_units	the letter spacing, in font units.
out	left	the left edge of the ink, in font units; untouched when no glyph draws.
out	right	the right edge of the ink, in font units; untouched when no glyph draws. NULL asks for the left edge alone, which lets the fold stop before the end of the string.

Returns. true when at least one glyph draws, false when none does.

2.6.3.4. VG_FREETYPE_string_width()

```
jfloat VG_FREETYPE_string_width (jchar * text, jint length, jint
face_handle, jfloat size, jfloat letter_spacing)
```

Measures the width of a text for the specified font and size.

The width is the union of the ink of every glyph, each at its own pen position: the leftmost pixel any glyph draws to the rightmost, whichever glyphs those are. A character that draws nothing contributes nothing, at either end.

Parameters .

Table 2.13.

in	text	the array of characters to draw.
in	length	the length of the array
in	face_handle	the font reference handle
in	size	the height of the font in pixels.
in	letter_spacing	the extra letter spacing to use

Returns. the width of the specified string, in pixels, never negative.

2.7. bsp/vg/inc/vg_helper.h File Reference

MicroEJ MicroVG library low level API: helper to implement library natives methods.

```
#include <sni.h>
#include "vg_configuration.h"
```

Macros

- `#define VG_LOG_ERROR(fmt, ...) VG_LOG_ERROR_PRINT(fmt, ## __VA_ARGS__)`
Logs an error. The format has no trailing line return.
- `#define VG_LOG_INFO(fmt, ...)`
Logs an informative message. The format has no trailing line return.
- `#define MEJ_LOG_ERROR_MICROVG(fmt, ...) VG_LOG_ERROR(fmt, ## __VA_ARGS__)`
Deprecated, use `VG_LOG_ERROR()`. Kept for the Abstraction Layers that build on this one.
- `#define MEJ_LOG_INFO_MICROVG(fmt, ...) VG_LOG_INFO(fmt, ## __VA_ARGS__)`
Deprecated, use `VG_LOG_INFO()`. Kept for the Abstraction Layers that build on this one.
- `#define VG_HELPER_NULL_GRADIENT 0`
Set this define to monitor freetype heap evolution. It needs `VG_LOG_INFO_ENABLED` to print the heap logs.
- `#define FT_FACE_FLAG_COMPLEX_LAYOUT (((uint32_t)1) << 31)`

Freetype supplementary flag for complex layout Uses a free bit in freetype face flags to convey the complex layout mode information with the freetype face. freetype.h must be checked on freetype update to ensure that this bit is still free.

- `#define 3.1415926535`
- `#define (r) ((r) * (180.0f / M_PI))`
- `#define (d) (((d) * M_PI) / 180.0f)`
- `#define (f) (*(uint32_t *)&(f))`
- `#define (i) (*(float *)&(i))`
- `#define REGISTERDESC(desc, buf, buf_len) if ((buf_len) >= sizeof(desc)) { (void)memcpy((buf), (desc), sizeof(desc)); }`

Registers the description of a SNI close function.

Functions

- `void LLUI_DISPLAY_waitAsynchronousDrawingEnd (void)`
Waits until the end of current asynchronous drawing.
- `int VG_HELPER_get_utf (const unsigned short *text, int length, int *offset)`
Gets the UTF character from a text buffer at the given offset and updates the offset to point to the next character.
- `void VG_HELPER_layout_configure (int faceHandle, const unsigned short *text, int length)`
Configures the font layouter with a font and a text.
- `void VG_HELPER_layout_dispose (int faceHandle)`
Releases the font layouter's state built from a font face, when that face is about to be freed. Does nothing when the layouter holds no state for this face.
- `bool VG_HELPER_layout_load_glyph (uint32_t *glyph_idx, int *x_advance, int *y_advance, int *x_offset, int *y_offset)`
Loads the next layouted glyph and gets index and positions.
- `void VG_HELPER_layout_stop (void)`
Releases what the layouter still holds for the current iteration. Does nothing when there is nothing to release, so a caller that may or may not have stopped early calls it either way, and calling it twice is safe.
- `const jfloat * VG_HELPER_check_matrix (const jfloat *matrix)`
Checks if the matrix is null. In that case, returns an identity matrix. This allows to prevent to make some checks on the matrix in the algorithms.
- `uint32_t VG_HELPER_apply_alpha (uint32_t color, uint32_t alpha)`

Applies the global opacity on given color.

- void VG_HELPER_prepare_matrix (jfloat *dest, jfloat x, jfloat y, const jfloat *matrix)

Converts a MicroVG matrix in another MicroVG matrix applying a translation.

2.7.1. Detailed Description

MicroEJ MicroVG library low level API: helper to implement library natives methods.

Author. MicroEJ Developer Team

Version. 8.0.3

2.7.2. Macro Definition Documentation

2.7.2.1. VG_HELPER_NULL_GRADIENT

```
#define VG_HELPER_NULL_GRADIENT 0
```

Set this define to monitor freetype heap evolution. It needs VG_LOG_INFO_ENABLED to print the heap logs.

NULL Gradient value

Definition at line 75 of file vg_helper.h.

2.7.2.2. VG_LOG_ERROR

```
#define VG_LOG_ERROR( fmt, ... ) VG_LOG_ERROR_PRINT( fmt, ## __VA_ARGS__ )
```

Logs an error. The format has no trailing line return.

The errors are always compiled. The printer is VG_LOG_ERROR_PRINT() (see vg_configuration.h).

Definition at line 42 of file vg_helper.h.

2.7.2.3. VG_LOG_INFO

```
#define VG_LOG_INFO( fmt, ... )
```

Logs an informative message. The format has no trailing line return.

Compiled only when VG_LOG_INFO_ENABLED is 1. The printer is VG_LOG_INFO_PRINT() (see vg_configuration.h).

Definition at line 53 of file vg_helper.h.

2.7.3. Function Documentation

2.7.3.1. LLUI_DISPLAY_waitAsynchronousDrawingEnd()

```
void LLUI_DISPLAY_waitAsynchronousDrawingEnd (void )
```

Waits until the end of current asynchronous drawing.

To avoid potential side effects from the release of objects (images, fonts) retained by a feature during the killing of that feature, ensure that no third-party components (e.g., GPU) are using these objects at the time of the kill.

UI packs with versions higher than 14.5.1 provide the blocking API `LLUI_DISPLAY_waitAsynchronousDrawingEnd()`. This API is stubbed on the VG Pack for backward compatibility issues between VG Pack 1.8.0 and UI Packs [14.4.0, 14.5.1]. However, it is highly recommended to use a UI Pack version greater than 14.5.1.

Definition at line 157 of file `vg_helper.c`.

```
157
    {
158
159 }
```

2.7.3.2. VG_HELPER_apply_alpha()

```
uint32_t VG_HELPER_apply_alpha (uint32_t color, uint32_t alpha)
```

Applies the global opacity on given color.

Parameters .

Table 2.14.

in	color	the 32-bit color.
in	alpha	the opacity.

Returns. the new color.

Definition at line 352 of file `vg_helper.c`.

```
352
353     uint32_t color_alpha = (((color >> 24) & (uint32_t)0xff) *
    alpha) / (uint32_t)255;
354     (color & (uint32_t)0xffffffff) | (color_alpha << 24);
355 }
```

2.7.3.3. VG_HELPER_check_matrix()

```
const jfloat* VG_HELPER_check_matrix (const jfloat * matrix)
```

Checks if the matrix is null. In that case, returns an identity matrix. This allows to prevent to make some checks on the matrix in the algorithms.

The identity matrix can be used several times by the same algorithm. The caller must not modify it (read-only matrix).

Parameters .

Table 2.15.

in	matrix	the matrix to check
----	--------	---------------------

Returns. the matrix or an identity matrix

Definition at line 347 of file `vg_helper.c`.

```
347                                     {  
348     (NULL == matrix) ? g_identity_matrix : matrix;  
349 }
```

2.7.3.4. VG_HELPER_get_utf()

```
int VG_HELPER_get_utf (const unsigned short * text, int length, int *  
offset)
```

Gets the UTF character from a text buffer at the given offset and updates the offset to point to the next character.

Some characters have some special values; they are made up of two Unicode characters in two specific ranges such that the first Unicode character is in one range (for example 0xD800-0xD8FF) and the second Unicode character is in the second range (for example 0xDC00-0xDCFF). This is called a surrogate pair.

If a surrogate pair is incomplete (missing second character), this function returns "0" (error) and does not update the offset.

Parameters .

Table 2.16.

in	text	text buffer encoded in UTF16 where to read UTF character.
in	length	length of the text buffer.
in,out	offset	offset in the text buffer where to read UTF character. Updated to the next character position.

Returns. The decoded UTF character.

Definition at line 166 of file `vg_helper.c`.

```

166      {
167          highPart = GET_NEXT_CHARACTER(textCharRam, length, *offset);
168          ret = 0;
169
170          ((highPart >= MIN_HIGH_SURROGATE) && (highPart <=
171             MAX_HIGH_SURROGATE)) {
172              (*offset < (length - 1)) {
173                  lowPart = GET_NEXT_CHARACTER(textCharRam, length,
174                      *(offset) + 1);
175
176                  ((lowPart >= MIN_LOW_SURROGATE) && (lowPart <=
177                     MAX_LOW_SURROGATE)) {
178                      *offset += 2;
179
180                      ret = 0;
181                      ret += ((int)highPart - ()MIN_HIGH_SURROGATE);
182                      ret <=<= (int)10;
183                      ret += ((int)lowPart - ()MIN_LOW_SURROGATE);
184                      ret += (int)MIN_SUPPLEMENTARY_CODE_POINT;
185                  }
186              }
187          } {
188              *offset += 1;
189
190              ret = 0x0000FFFF & (int)highPart;
191          }
192
193          ret;
194 }

```

2.7.3.5. VG_HELPER_layout_configure()

```
void VG_HELPER_layout_configure (int faceHandle, const unsigned short
* text, int length)
```

Configures the font layouter with a font and a text.

Releases what an iteration left open, as VG_HELPER_layout_stop() does, so a caller that stops before the end of its own iteration costs the next caller nothing.

Parameters .

Table 2.17.

in	faceHandle	handle on font face.
in	text	text buffer encoded in UTF16 where to read UTF character.
in	length	text buffer length.

2.7.3.6. VG_HELPER_layout_dispose()

```
void VG_HELPER_layout_dispose (int faceHandle)
```

Releases the font layouter's state built from a font face, when that face is about to be freed. Does nothing when the layouter holds no state for this face.

Must be called before the face itself is freed: the state holds the face and reads through it for its whole life.

Parameters .

Table 2.18.

in	faceHandle	handle on the font face being freed.
----	------------	--------------------------------------

2.7.3.7. VG_HELPER_layout_load_glyph()

```
bool VG_HELPER_layout_load_glyph (uint32_t * glyph_idx, int * x_advance,  
int * y_advance, int * x_offset, int * y_offset)
```

Loads the next layouted glyph and gets index and positions.

Parameters .

Table 2.19.

out	glyph_idx	next glyph index.
out	x_advance	the horizontal advance to add to the cursor position after drawing the glyph.
out	y_advance	the vertical advance to add to the cursor after drawing the glyph.
out	x_offset	the horizontal offset of the glyph, does not affect the cursor position.
out	y_offset	the vertical offset of the glyph, does not affect the cursor position.

Returns. true if a glyph is available otherwise false.

2.7.3.8. VG_HELPER_layout_stop()

```
void VG_HELPER_layout_stop (void )
```

Releases what the layouter still holds for the current iteration. Does nothing when there is nothing to release, so a caller that may or may not have stopped early calls it either way, and calling it twice is safe.

It does not reset the reading position, so the next iteration starts with VG_HELPER_layout_configure() as it always did.

2.7.3.9. VG_HELPER_prepare_matrix()

```
void VG_HELPER_prepare_matrix (jfloat * dest, jfloat x, jfloat y, const
jfloat * matrix)
```

Converts a MicroVG matrix in another MicroVG matrix applying a translation.

Parameters .

Table 2.20.

out	dest	the MicroVG matrix to set
in	x	the X translation to apply
in	y	the Y translation to apply
in	matrix	the MicroVG transformation to apply (can be null)

Definition at line 358 of file vg_helper.c.

```
358
359         {
360     jfloat *local_matrix = VG_HELPER_check_matrix(matrix);
361     ((0 != x) || (0 != y)) {
362         LLVG_MATRIX_IMPL_setTranslate(dest, x, y);
363         LLVG_MATRIX_IMPL_concatenate(dest, local_matrix);
364     } {
365         LLVG_MATRIX_IMPL_copy(dest, local_matrix);
366     }
367 }
368 }
369 }
```

2.8. bsp/vg/inc/vg_outline_box.h File Reference

MicroEJ MicroVG library low level API: box of the curve an outline draws.

```
#include "vg_configuration.h"
#include <stdbool.h>
#include <freetype/ftoutln.h>
```

Data Structures

- struct VG_OUTLINE_BOX

The box of the curve an outline draws, control points excluded.

Functions

- void VG_OUTLINE_BOX_compute (const FT_Outline *outline, VG_OUTLINE_BOX *box)

Computes the box of the curve the given outline draws, where the box a font declares also bounds the control points, which the curve never touches.

2.8.1. Detailed Description

MicroEJ MicroVG library low level API: box of the curve an outline draws.

Author. MicroEJ Developer Team

Version. 8.0.3

2.8.2. Function Documentation

2.8.2.1. VG_OUTLINE_BOX_compute()

```
void VG_OUTLINE_BOX_compute (const FT_Outline * outline, VG_OUTLINE_BOX
* box)
```

Computes the box of the curve the given outline draws, where the box a font declares also bounds the control points, which the curve never touches.

The walk relies on FT_Outline_Decompose, which validates nothing: it reports only the malformations it happens to trip over, and an outline whose contour ends past its point array is read past that array and yields an arbitrary box. Passing a well-formed outline is the caller's responsibility; every caller passes an outline FreeType itself produced.

Parameters .

Table 2.21.

in	outline	the well-formed outline to measure
out	box	the box, zeroed and flagged empty when the outline draws nothing, and when FreeType reports a malformation while walking it

2.9. bsp/vg/inc/vg_path.h File Reference

MicroEJ MicroVG library low level API: implementation of Path.

```
#include "vg_configuration.h"
#include <sni.h>
#include <stddef.h>
```

Data Structures

- struct MICROVG_PATH_HEADER

Map a jbyte array that represents a path.

Typedefs

- typedef struct MICROVG_PATH_HEADER VG_PATH_HEADER_t

Map a jbyte array that represents a path.

Functions

- uint8_t VG_PATH_get_path_encoder_format (void)

Gets the path's array format used to encode the commands and parameters. This format is stored into the path's header.

- uint32_t VG_PATH_convert_path_command (jint command)

Converts the command in destination format.

- void VG_PATH_initialize (void)

Initializes the path builder.

- uint32_t VG_PATH_get_path_header_size (void)

Gets the path's array header size.

- uint32_t VG_PATH_get_path_command_size (jint command, uint32_t nbParams)

Gets the size to add in the path array to encode the command and its parameters.

- uint32_t VG_PATH_append_path_command0 (jbyte *path, jint array_length, jint cmd)

Appends the command with zero parameter in the path's array.

- uint32_t VG_PATH_append_path_command1 (jbyte *path, jint array_length, jint cmd, jfloat x, jfloat y)

Appends the command with 1 point parameter in the path's array.

- uint32_t VG_PATH_append_path_command2 (jbyte *path, jint array_length, jint cmd, jfloat x1, jfloat y1, jfloat x2, jfloat y2)

Appends the command with 2 points parameter in the path's array.

- uint32_t VG_PATH_append_path_command3 (jbyte *path, jint array_length, jint cmd, jfloat x1, jfloat y1, jfloat x2, jfloat y2, jfloat x3, jfloat y3)

Appends the command with 3 points parameter in the path's array.

- uint32_t VG_PATH_get_command_parameter_number (jint command)

Gets the number of parameters for a specific command.

2.9.1. Detailed Description

MicroEJ MicroVG library low level API: implementation of Path.

Author. MicroEJ Developer Team

Version. 8.0.3

2.9.2. Function Documentation

2.9.2.1. VG_PATH_append_path_command0()

```
uint32_t VG_PATH_append_path_command0 (jbyte * path, jint array_length,
jint cmd)
```

Appends the command with zero parameter in the path's array.

The caller ensures the path's array is large enough to encode the command and its parameters.

The default implementation uses 32-bit fields for the command and for each data.

Parameters .

Table 2.22.

in	path	the path's array
in	array_length	the length of the array
in	cmd	the command to add

Returns. LLVG_SUCCESS on a success, or the missing space in the array in bytes

See also. VG_PATH_get_path_command_size(jint, uint32_t)

2.9.2.2. VG_PATH_append_path_command1()

```
uint32_t VG_PATH_append_path_command1 (jbyte * path, jint array_length,
jint cmd, jfloat x, jfloat y)
```

Appends the command with 1 point parameter in the path's array.

Parameters .

Table 2.23.

in	path	the path's array
in	array_length	the length of the array
in	cmd	the command to add
in	x	the command data 1.
in	y	the command data 2.

Returns. LLVG_SUCCESS on a success, or the missing space in the array in bytes

See also. VG_PATH_append_path_command0()

2.9.2.3. VG_PATH_append_path_command2()

```
uint32_t VG_PATH_append_path_command2 (jbyte * path, jint array_length,  
jint cmd, jfloat x1, jfloat y1, jfloat x2, jfloat y2)
```

Appends the command with 2 points parameter in the path's array.

Parameters .

Table 2.24.

in	path	the path's array
in	array_length	the length of the array
in	cmd	the command to add
in	x1	the command data 1.
in	y1	the command data 2.
in	x2	the command data 3.
in	y2	the command data 4.

Returns. LLVG_SUCCESS on a success, or the missing space in the array in bytes

See also. VG_PATH_append_path_command0()

2.9.2.4. VG_PATH_append_path_command3()

```
uint32_t VG_PATH_append_path_command3 (jbyte * path, jint array_length,  
jint cmd, jfloat x1, jfloat y1, jfloat x2, jfloat y2, jfloat x3, jfloat  
y3)
```

Appends the command with 3 points parameter in the path's array.

Parameters .

Table 2.25.

in	path	the path's array
in	array_length	the length of the array
in	cmd	the command to add
in	x1	the command data 1.
in	y1	the command data 2.
in	x2	the command data 3.
in	y2	the command data 4.
in	x3	the command data 5.
in	y3	the command data 6.

Returns. LLVG_SUCCESS on a success, or the missing space in the array in bytes

See also. `VG_PATH_append_path_command0()`

2.9.2.5. `VG_PATH_convert_path_command()`

```
uint32_t VG_PATH_convert_path_command (jint command)
```

Converts the command in destination format.

Parameters .

Table 2.26.

in	command	the command to convert
----	---------	------------------------

Returns. the converted command

2.9.2.6. `VG_PATH_get_command_parameter_number()`

```
uint32_t VG_PATH_get_command_parameter_number (jint command)
```

Gets the number of parameters for a specific command.

Parameters .

Table 2.27.

in	command	the command.
----	---------	--------------

Returns. the number of parameters for the given command.

2.9.2.7. `VG_PATH_get_path_command_size()`

```
uint32_t VG_PATH_get_path_command_size (jint command, uint32_t nbParams)
```

Gets the size to add in the path array to encode the command and its parameters.

The default implementation uses 32-bit fields for the command and for each data.

Parameters .

Table 2.28.

in	command	the command to add.
in	nbParams	the available number of command's parameters

Returns. the size to add in the path array.

2.9.2.8. `VG_PATH_get_path_encoder_format()`

```
uint8_t VG_PATH_get_path_encoder_format (void )
```

Gets the path's array format used to encode the commands and parameters. This format is stored into the path's header.

Returns. the path's format

2.9.2.9. VG_PATH_get_path_header_size()

```
uint32_t VG_PATH_get_path_header_size (void )
```

Gets the path's array header size.

The default implementation returns sizeof(VG_PATH_HEADER_t).

Returns. the size in bytes.

2.9.2.10. VG_PATH_initialize()

```
void VG_PATH_initialize (void )
```

Initializes the path builder.

The default implementation does nothing.

2.10. bsp/vg/src/LLVG_BVI_impl_stub.c File Reference

MicroVG library low level API: stubbed implementation of LLVG_BVI_impl.h.

```
#include "vg_configuration.h"
```

2.10.1. Detailed Description

MicroVG library low level API: stubbed implementation of LLVG_BVI_impl.h.

Author. MicroEJ Developer Team

Version. 8.0.3

2.11. bsp/vg/src/LLVG_FONT_impl_freetype.c File Reference

MicroEJ MicroVG library low level API: implementation over FreeType.

```
#include <assert.h>
#include "vg_configuration.h"
```

2.11.1. Detailed Description

MicroEJ MicroVG library low level API: implementation over FreeType.

Author. MicroEJ Developer Team

Version. 8.0.3

2.12. bsp/vg/src/LLVG_FONT_impl_stub.c File Reference

MicroEJ MicroVG library low level API: stub implementation of LLVG_FONT_IMPL_xxx().

```
#include "vg_configuration.h"
#include <LLVG_FONT_impl.h>
#include <LLVG_PAINTER_impl.h>
```

Functions

- jint (jchar *font_name, jboolean complex_layout)
- jfloat (jchar *text, jint faceHandle, jfloat size, jfloat letterSpacing)
- jfloat (jchar *text, jint faceHandle, jfloat size)
- jfloat (jint faceHandle, jfloat size)
- jfloat (jint faceHandle, jfloat size)
- void (jint faceHandle)
- bool (void)

2.12.1. Detailed Description

MicroEJ MicroVG library low level API: stub implementation of LLVG_FONT_IMPL_xxx().

Author. MicroEJ Developer Team

Version. 8.0.3

2.13. bsp/vg/src/LLVG_GRADIENT_impl_stub.c File Reference

MicroVG library low level API: stub of the linear gradient implementation.

```
#include "vg_configuration.h"
```

2.13.1. Detailed Description

MicroVG library low level API: stub of the linear gradient implementation.

This implementation is only compatible with the configuration VG_FEATURE_GRADIENT_FIRST_COLOR. It only stores the very first gradient's color as current color.

Author. MicroEJ Developer Team

Version. 8.0.3

2.14. bsp/vg/src/LLVG_MATRIX_impl.c File Reference

MicroEJ MicroVG library low level API: basic implementation of matrix APIs.

```
#include <math.h>
#include <string.h>
#include <LLVG_MATRIX_impl.h>
#include "vg_helper.h"
```

Functions

- void (jfloat *matrix)
- void (jfloat *dest, const jfloat *src)
- void (jfloat *dest, const jfloat *a, const jfloat *b)
- void (jfloat *matrix, jfloat x, jfloat y)
- void (jfloat *matrix, jfloat sx, jfloat sy)
- void (jfloat *matrix, jfloat degrees)
- void (jfloat *dest, const jfloat *a, const jfloat *b)
- void (jfloat *matrix, jfloat x, jfloat y)
- void (jfloat *matrix, jfloat scaleX, jfloat scaleY)
- void (jfloat *matrix, jfloat angleDegrees)
- void (jfloat *matrix, const jfloat *other)
- void (jfloat *matrix, jfloat dx, jfloat dy)
- void (jfloat *matrix, jfloat sx, jfloat sy)
- void (jfloat *matrix, jfloat degrees)
- void (jfloat *matrix, const jfloat *other)

- void (jfloat *x, jfloat *y, const jfloat *matrix)

2.14.1. Detailed Description

MicroEJ MicroVG library low level API: basic implementation of matrix APIs.

Author. MicroEJ Developer Team

Version. 8.0.3

2.15. bsp/vg/src/LLVG_PAINTER_impl.c File Reference

This file implements all MicroVG drawing native functions.

```
#include <LLVG_PAINTER_impl.h>
#include <LLUI_DISPLAY.h>
#include <LLVG_FONT_impl.h>
#include <LLVG_MATRIX_impl.h>
#include "vg_drawing.h"
#include "vg_helper.h"
#include "vg_trace.h"
```

Functions

- jint (MICROUI_GraphicsContext *gc, jbyte *pathData, jint x, jint y, jfloat *matrix, jint fillRule, jint blend, jint color)
- jint (MICROUI_GraphicsContext *gc, jbyte *pathData, jint x, jint y, jfloat *matrix, jint fillRule, jint alpha, jint blend, jint *gradientData, jfloat *gradientMatrix)
- jint (MICROUI_GraphicsContext *gc, jchar *text, jint faceHandle, jfloat size, jfloat x, jfloat y, jfloat *matrix, jint alpha, jint blend, jfloat letterSpacing)
- jint (MICROUI_GraphicsContext *gc, jchar *text, jint faceHandle, jfloat size, jfloat x, jfloat y, jfloat *matrix, jint alpha, jint blend, jfloat letterSpacing, jint *gradientData, jfloat *gradientMatrix)
- jint (MICROUI_GraphicsContext *gc, jchar *text, jint faceHandle, jfloat size, jint x, jint y, jfloat *matrix, jint alpha, jint blend, jfloat letterSpacing, jfloat radius, jint direction)
- jint (MICROUI_GraphicsContext *gc, jchar *text, jint faceHandle, jfloat size, jint x, jint y, jfloat *matrix, jint alpha, jint blend, jfloat letterSpacing, jfloat radius, jint direction, jint *gradientData, jfloat *gradientMatrix)
- jint (MICROUI_GraphicsContext *gc, MICROVG_Image *image, jint x, jint y, jfloat *matrix, jint alpha, jlong elapsed, const float color_matrix[])

2.15.1. Detailed Description

This file implements all MicroVG drawing native functions.

See also. LLVG_PAINTER_impl.h file comment

Author. MicroEJ Developer Team

Version. 8.0.3

2.16. bsp/vg/src/LLVG_PATH_impl_dual.c File Reference

MicroVG library low level API: implementation of path.

```
#include "vg_configuration.h"
```

2.16.1. Detailed Description

MicroVG library low level API: implementation of path.

This implementation uses a 32-bit "integer" value to store a path command and a 32-bit "float" value to store each command parameter.

The encoding can be overridden, see "[optional]: weak functions" in "vg_path.h"

Author. MicroEJ Developer Team

Version. 8.0.3

2.17. bsp/vg/src/LLVG_PATH_impl_single.c File Reference

MicroVG library low level API: implementation of path.

```
#include "vg_configuration.h"
```

2.17.1. Detailed Description

MicroVG library low level API: implementation of path.

This implementation uses a 32-bit "integer" value to store a path command and a 32-bit "float" value to store each command parameter.

The encoding can be overridden, see "[optional]: weak functions" in "vg_path.h"

Author. MicroEJ Developer Team

Version. 8.0.3

2.18. bsp/vg/src/LLVG_PATH_impl_stub.c File Reference

MicroVG library low level API: stubbed implementation of LLVG_PATH_impl.h and LLVG_PATH_PAINTER_impl.h.

```
#include "vg_configuration.h"
#include <LLVG_PATH_impl.h>
```

Functions

- jint (jbyte *jpath, jint length)
- jint (jbyte *jpath, jint length, jint cmd, jfloat x, jfloat y)
- jint (jbyte *jpath, jint length, jint cmd, jfloat x1, jfloat y1, jfloat x2, jfloat y2)
- jint (jbyte *jpath, jint length, jint cmd, jfloat x1, jfloat y1, jfloat x2, jfloat y2, jfloat x3, jfloat y3)
- void (jbyte *jpath)

2.18.1. Detailed Description

MicroVG library low level API: stubbed implementation of LLVG_PATH_impl.h and LLVG_PATH_PAINTER_impl.h.

Author. MicroEJ Developer Team

Version. 8.0.3

2.19. bsp/vg/src/vg_freetype_path.c File Reference

MicroEJ MicroVG library low level API: implementation over Freetype.

```
#include "vg_configuration.h"
```

2.19.1. Detailed Description

MicroEJ MicroVG library low level API: implementation over Freetype.

Author. MicroEJ Developer Team

Version. 8.0.3

2.20. bsp/vg/src/vg_helper.c File Reference

MicroEJ MicroVG library low level API: helper to implement library natives methods.

```
#include "vg_configuration.h"
#include <LLVG_MATRIX_impl.h>
#include "vg_helper.h"
#include "vg_trace.h"
#include "vg_drawing.h"
#include "bsp_util.h"
```

Macros

- #define ((unsigned short)0xD800)
- #define ((unsigned short)0xDBFF)
- #define ((unsigned short)0xDC00)
- #define ((unsigned short)0xDFFF)
- #define 0x010000
- #define (t, l, o) ((o) >= (l) ? (unsigned short)0 : (t)[o])
- #define true

Functions

- void (jint trace_group)
- BSP_DECLARE_WEAK_FCNT void LLUI_DISPLAY_waitAsynchronousDrawingEnd (void)
Waits until the end of current asynchronous drawing.
- int VG_HELPER_get_utf (const unsigned short *textCharRam, int length, int *offset)
Gets the UTF character from a text buffer at the given offset and updates the offset to point to the next character.
- const jfloat * VG_HELPER_check_matrix (const jfloat *matrix)
Checks if the matrix is null. In that case, returns an identity matrix. This allows to prevent to make some checks on the matrix in the algorithms.
- uint32_t VG_HELPER_apply_alpha (uint32_t color, uint32_t alpha)
Applies the global opacity on given color.
- void VG_HELPER_prepare_matrix (jfloat *dest, jfloat x, jfloat y, const jfloat *matrix)
Converts a MicroVG matrix in another MicroVG matrix applying a translation.

Variables

- int32_t LLVG_TRACE_group

Identifies the MicroVG group to trace an event.

2.20.1. Detailed Description

MicroEJ MicroVG library low level API: helper to implement library natives methods.

Author. MicroEJ Developer Team

Version. 8.0.3

2.20.2. Function Documentation

2.20.2.1. LLUI_DISPLAY_waitAsynchronousDrawingEnd()

```
BSP_DECLARE_WEAK_FCNT    void    LLUI_DISPLAY_waitAsynchronousDrawingEnd
(void )
```

Waits until the end of current asynchronous drawing.

To avoid potential side effects from the release of objects (images, fonts) retained by a feature during the killing of that feature, ensure that no third-party components (e.g., GPU) are using these objects at the time of the kill.

UI packs with versions higher than 14.5.1 provide the blocking API `LLUI_DISPLAY_waitAsynchronousDrawingEnd( )`. This API is stubbed on the VG Pack for backward compatibility issues between VG Pack 1.8.0 and UI Packs [14.4.0, 14.5.1]. However, it is highly recommended to use a UI Pack version greater than 14.5.1.

Definition at line 157 of file `vg_helper.c`.

```
157
    {
158
159 }
```

2.20.2.2. VG_HELPER_apply_alpha()

```
uint32_t VG_HELPER_apply_alpha (uint32_t color, uint32_t alpha)
```

Applies the global opacity on given color.

Parameters .

Table 2.29.

in	color	the 32-bit color.
----	-------	-------------------

in	alpha	the opacity.
----	-------	--------------

Returns. the new color.

Definition at line 352 of file vg_helper.c.

```

352
353     uint32_t color_alpha = (((color >> 24) & (uint32_t)0xff) *
alpha) / (uint32_t)255;
354     (color & (uint32_t)0xffffffff) | (color_alpha << 24);
355 }
```

2.20.2.3. VG_HELPER_check_matrix()

```
const jfloat* VG_HELPER_check_matrix (const jfloat * matrix)
```

Checks if the matrix is null. In that case, returns an identity matrix. This allows to prevent to make some checks on the matrix in the algorithms.

The identity matrix can be used several times by the same algorithm. The caller must not modify it (read-only matrix).

Parameters .

Table 2.30.

in	matrix	the matrix to check
----	--------	---------------------

Returns. the matrix or an identity matrix

Definition at line 347 of file vg_helper.c.

```

347
348     (NULL == matrix) ? g_identity_matrix : matrix;
349 }
```

2.20.2.4. VG_HELPER_get_utf()

```
int VG_HELPER_get_utf (const unsigned short * text, int length, int *
offset)
```

Gets the UTF character from a text buffer at the given offset and updates the offset to point to the next character.

Some characters have some special values; they are made up of two Unicode characters in two specific ranges such that the first Unicode character is in one range (for example 0xD800-0xD8FF) and the second Unicode character is in the second range (for example 0xDC00-0xDCFF). This is called a surrogate pair.

If a surrogate pair is incomplete (missing second character), this function returns "0" (error) and does not update the offset.

Parameters .

Table 2.31.

in	text	text buffer encoded in UTF16 where to read UTF character.
in	length	length of the text buffer.
in,out	offset	offset in the text buffer where to read UTF character. Updated to the next character position.

Returns. The decoded UTF character.

Definition at line 166 of file vg_helper.c.

```

166
167     {
168         highPart = GET_NEXT_CHARACTER(textCharRam, length, *offset);
169         ret = 0;
170         ((highPart >= MIN_HIGH_SURROGATE) && (highPart <=
171             MAX_HIGH_SURROGATE)) {
172             (*offset < (length - 1)) {
173                 lowPart = GET_NEXT_CHARACTER(textCharRam, length,
174                     *(offset) + 1);
175                 ((lowPart >= MIN_LOW_SURROGATE) && (lowPart <=
176                     MAX_LOW_SURROGATE)) {
177                     *offset += 2;
178                     ret = 0;
179                     ret += ((int)highPart - (int)MIN_HIGH_SURROGATE);
180                     ret <<= (int)10;
181                     ret += ((int)lowPart - (int)MIN_LOW_SURROGATE);
182                     ret += (int)MIN_SUPPLEMENTARY_CODE_POINT;
183                 }
184             }
185         } {
186             *offset += 1;
187         }
188         ret = 0x0000FFFF & (int)highPart;
189     }
190     ret;
191 }

```

2.20.2.5. VG_HELPER_prepare_matrix()

```
void VG_HELPER_prepare_matrix (jfloat * dest, jfloat x, jfloat y, const
jfloat * matrix)
```

Converts a MicroVG matrix in another MicroVG matrix applying a translation.

Parameters .

Table 2.32.

out	dest	the MicroVG matrix to set
in	x	the X translation to apply
in	y	the Y translation to apply
in	matrix	the MicroVG transformation to apply (can be null)

Definition at line 358 of file `vg_helper.c`.

```
358
359         {
360     jfloat *local_matrix = VG_HELPER_check_matrix(matrix);
361     ((0 != x) || (0 != y)) {
362         LLVG_MATRIX_IMPL_setTranslate(dest, x, y);
363         LLVG_MATRIX_IMPL_concatenate(dest, local_matrix);
364     } {
365         LLVG_MATRIX_IMPL_copy(dest, local_matrix);
366     }
367 }
368
369 }
```

2.21. bsp/vg/src/vg_outline_box.c File Reference

MicroEJ MicroVG library low level API: box of the curve an outline draws.

```
#include "vg_configuration.h"
```

2.21.1. Detailed Description

MicroEJ MicroVG library low level API: box of the curve an outline draws.

Author. MicroEJ Developer Team

Version. 8.0.3

Index

B

- bounds_xmax
 - MICROVG_PATH_HEADER, 2
- bounds_xmin
 - MICROVG_PATH_HEADER, 2
- bounds_ymax
 - MICROVG_PATH_HEADER, 2
- bounds_ymin
 - MICROVG_PATH_HEADER, 2
- bsp/vg/inc/ui_drawing_bvi.h, 6
- bsp/vg/inc/vg_configuration.h, 6
- bsp/vg/inc/vg_drawing.h, 11
- bsp/vg/inc/vg_drawing_bvi.h, 17
- bsp/vg/inc/vg_drawing_stub.h, 17
- bsp/vg/inc/vg_freetype.h, 18
- bsp/vg/inc/vg_helper.h, 22
- bsp/vg/inc/vg_outline_box.h, 29
- bsp/vg/inc/vg_path.h, 30
- bsp/vg/src/LLVG_BVI_impl_stub.c, 35
- bsp/vg/src/LLVG_FONT_impl_freetype.c, 35
- bsp/vg/src/LLVG_FONT_impl_stub.c, 36
- bsp/vg/src/LLVG_GRADIENT_impl_stub.c, 36
- bsp/vg/src/LLVG_MATRIX_impl.c, 37
- bsp/vg/src/LLVG_PAINTER_impl.c, 38
- bsp/vg/src/LLVG_PATH_impl_dual.c, 39
- bsp/vg/src/LLVG_PATH_impl_single.c, 39
- bsp/vg/src/LLVG_PATH_impl_stub.c, 40
- bsp/vg/src/vg_freetype_path.c, 40
- bsp/vg/src/vg_helper.c, 40
- bsp/vg/src/vg_outline_box.c, 45

D

- data_size
 - MICROVG_PATH_HEADER, 2

E

- empty
 - VG_OUTLINE_BOX, 4

L

- LLUI_DISPLAY_waitAsynchronousDrawingEnd
 - vg_helper.c, 42
 - vg_helper.h, 25

M

- MICROUI_FontVG, 1
- MICROVG_PATH_HEADER, 1
 - bounds_xmax, 2
 - bounds_xmin, 2
 - bounds_ymax, 2
 - bounds_ymin, 2
 - data_size, 3
 - padding1, 3

- padding2, 3
- padding3, 3

P

- padding1
 - MICROVG_PATH_HEADER, 3
- padding2
 - MICROVG_PATH_HEADER, 3
- padding3
 - MICROVG_PATH_HEADER, 3

V

- vg_configuration.h
 - VG_FEATURE_FONT, 8
 - VG_FEATURE_FONT_COMPLEX_LAYOUT, 8
 - VG_FEATURE_FONT_EXTERNAL, 8
 - VG_FEATURE_FONT_FREETYPE_BITMAP, 9
 - VG_FEATURE_FONT_FREETYPE_VECTOR, 9
 - VG_FEATURE_FREETYPE_HEAP_SIZE, 9
 - VG_FEATURE_GRADIENT, 9
 - VG_FEATURE_GRADIENT_FIRST_COLOR, 10
 - VG_FEATURE_GRADIENT_FULL, 10
 - VG_FEATURE_PATH, 10
 - VG_LOG_DECLARE_MODULE, 10
 - VG_LOG_INFO_ENABLED, 10
- vg_drawing.h
 - VG_DRAWING_drawGradient, 12
 - VG_DRAWING_drawImage, 13
 - VG_DRAWING_drawPath, 13
 - VG_DRAWING_drawString, 14
 - VG_DRAWING_drawStringGradient, 14
 - VG_DRAWING_drawStringOnCircle, 15
 - VG_DRAWING_drawStringOnCircleGradient, 15
 - VG_DRAWING_get_image_size, 16
 - VG_DRAWING_image_is_closed, 16
 - VG_DRAWING_initialize, 17
- VG_DRAWING_drawGradient
 - vg_drawing.h, 12
- VG_DRAWING_drawImage
 - vg_drawing.h, 13
- VG_DRAWING_drawPath
 - vg_drawing.h, 13
- VG_DRAWING_drawString
 - vg_drawing.h, 14
- VG_DRAWING_drawStringGradient
 - vg_drawing.h, 14
- VG_DRAWING_drawStringOnCircle
 - vg_drawing.h, 15
- VG_DRAWING_drawStringOnCircleGradient
 - vg_drawing.h, 15
- VG_DRAWING_get_image_size
 - vg_drawing.h, 16
- VG_DRAWING_image_is_closed
 - vg_drawing.h, 16
- VG_DRAWING_initialize
 - vg_drawing.h, 17
- VG_FEATURE_FONT

- vg_configuration.h, 8
- VG_FEATURE_FONT_COMPLEX_LAYOUT
 - vg_configuration.h, 8
- VG_FEATURE_FONT_EXTERNAL
 - vg_configuration.h, 8
- VG_FEATURE_FONT_FREETYPE_BITMAP
 - vg_configuration.h, 9
- VG_FEATURE_FONT_FREETYPE_VECTOR
 - vg_configuration.h, 9
- VG_FEATURE_FREETYPE_HEAP_SIZE
 - vg_configuration.h, 9
- VG_FEATURE_GRADIENT
 - vg_configuration.h, 9
- VG_FEATURE_GRADIENT_FIRST_COLOR
 - vg_configuration.h, 10
- VG_FEATURE_GRADIENT_FULL
 - vg_configuration.h, 10
- VG_FEATURE_PATH
 - vg_configuration.h, 10
- vg_freetype.h
 - VG_FREETYPE_draw_glyph_t, 19
 - VG_FREETYPE_draw_string, 20
 - VG_FREETYPE_get_scale, 20
 - VG_FREETYPE_string_span, 21
 - VG_FREETYPE_string_width, 21
- VG_FREETYPE_draw_glyph_t
 - vg_freetype.h, 19
- VG_FREETYPE_draw_string
 - vg_freetype.h, 20
- VG_FREETYPE_get_scale
 - vg_freetype.h, 20
- VG_FREETYPE_string_span
 - vg_freetype.h, 21
- VG_FREETYPE_string_width
 - vg_freetype.h, 21
- vg_helper.c
 - LLUI_DISPLAY_waitAsynchronousDrawingEnd, 42
 - VG_HELPER_apply_alpha, 42
 - VG_HELPER_check_matrix, 43
 - VG_HELPER_get_utf, 43
 - VG_HELPER_prepare_matrix, 45
- vg_helper.h
 - LLUI_DISPLAY_waitAsynchronousDrawingEnd, 25
 - VG_HELPER_apply_alpha, 25
 - VG_HELPER_check_matrix, 25
 - VG_HELPER_get_utf, 26
 - VG_HELPER_layout_configure, 27
 - VG_HELPER_layout_dispose, 28
 - VG_HELPER_layout_load_glyph, 28
 - VG_HELPER_layout_stop, 28
 - VG_HELPER_NULL_GRADIENT, 24
 - VG_HELPER_prepare_matrix, 28
 - VG_LOG_ERROR, 24
 - VG_LOG_INFO, 24
- VG_HELPER_apply_alpha
 - vg_helper.c, 42
 - vg_helper.h, 25
- VG_HELPER_check_matrix
 - vg_helper.c, 43
 - vg_helper.h, 25
- VG_HELPER_get_utf
 - vg_helper.c, 43
 - vg_helper.h, 26
- VG_HELPER_layout_configure
 - vg_helper.h, 27
- VG_HELPER_layout_dispose
 - vg_helper.h, 28
- VG_HELPER_layout_load_glyph
 - vg_helper.h, 28
- VG_HELPER_layout_stop
 - vg_helper.h, 28
- VG_HELPER_NULL_GRADIENT
 - vg_helper.h, 24
- VG_HELPER_prepare_matrix
 - vg_helper.c, 44
 - vg_helper.h, 28
- VG_LOG_DECLARE_MODULE
 - vg_configuration.h, 10
- VG_LOG_ERROR
 - vg_helper.h, 24
- VG_LOG_INFO
 - vg_helper.h, 24
- VG_LOG_INFO_ENABLED
 - vg_configuration.h, 10
- VG_OUTLINE_BOX, 3
 - empty, 4
 - x_max, 4
 - x_min, 4
 - y_max, 4
 - y_min, 4
- vg_outline_box.h
 - VG_OUTLINE_BOX_compute, 30
- VG_OUTLINE_BOX_compute
 - vg_outline_box.h, 30
- vg_path.h
 - VG_PATH_append_path_command0, 32
 - VG_PATH_append_path_command1, 32
 - VG_PATH_append_path_command2, 33
 - VG_PATH_append_path_command3, 33
 - VG_PATH_convert_path_command, 34
 - VG_PATH_get_command_parameter_number, 34
 - VG_PATH_get_path_command_size, 34
 - VG_PATH_get_path_encoder_format, 34
 - VG_PATH_get_path_header_size, 35
 - VG_PATH_initialize, 35
- VG_PATH_append_path_command0
 - vg_path.h, 32
- VG_PATH_append_path_command1
 - vg_path.h, 32
- VG_PATH_append_path_command2
 - vg_path.h, 33
- VG_PATH_append_path_command3
 - vg_path.h, 33
- VG_PATH_convert_path_command
 - vg_path.h, 34
- VG_PATH_get_command_parameter_number
 - vg_path.h, 34

vg_path.h, 34
VG_PATH_get_path_command_size
vg_path.h, 34
VG_PATH_get_path_encoder_format
vg_path.h, 34
VG_PATH_get_path_header_size
vg_path.h, 35
VG_PATH_initialize
vg_path.h, 35

X

x_max
VG_OUTLINE_BOX, 4
x_min
VG_OUTLINE_BOX, 4

Y

y_max
VG_OUTLINE_BOX, 4
y_min
VG_OUTLINE_BOX, 4